**PRELIMINARY REPORT**

**Lab 06**

MUHAMMAD ARHAM KHAN

21701848 - CS

SECTION 06

SPRING 2019

**Dated: 05 May, 2019**

**Question 1:**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| No. | Cache  Size KB | N way  cache | Word  Size | Block size  (no. of words) | No. of  Sets | Tag Size  in bits | Index Size  (Set No.) in bits | Word Block  Offset  Size in bits1 | Byte  Offset  Size in bits2 | Block  Replacement  Policy Needed (Yes/No) |
| 1 | 32 | 1 | 32 bits | 4 | **2048** | **14** | **11** | **2** | **2** | **No** |
| 2 | 32 | 2 | 32 bits | 4 | **1024** | **15** | **10** | **2** | **2** | **Yes** |
| 3 | 32 | 4 | 32 bits | 8 | **256** | **16** | **8** | **3** | **2** | **Yes** |
| 4 | 32 | Full | 32 bits | 8 | **1** | **24** | **0** | **3** | **2** | **Yes** |
| 9 | 256 | 1 | 16 bits | 4 | **32768** | **11** | **15** | **2** | **1** | **No** |
| 10 | 256 | 2 | 16 bits | 4 | **16384** | **12** | **14** | **2** | **1** | **Yes** |
| 11 | 256 | 4 | 8 bits | 16 | **4096** | **13** | **12** | **4** | **0** | **Yes** |
| 12 | 256 | Full | 8 bits | 16 | **1** | **25** | **0** | **4** | **0** | **Yes** |

**1** Word Block Offset Size in bits: Log2(No. of words in a block)

**2** Byte Offset Size in bits: Log2(No. of bytes in a word)

**Question 2:**

**Part a:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Instruction | Iteration No. | | | | | |
| **1** | **2** | **3** | **4** | **5** |
| lw $t1, 0x24($0) | Compulsory |  |  |  |  |
| lw $t2, 0x2C($0) | Compulsory |  |  |  |  |
| lw $t3, 0x28($0) |  |  |  |  |  |

**Part b:**

Since cache contains 4 sets, so there are 2 set bits, 2 byte offsets, 1 bit block offset, 27 tag bits (Memory address)

The cache memory stores the tag bits, V bits and a 32 bit memory.

So, cache size per size = 1 (V) + 27 (Tag) + 32 (Word 1) + 32 (Word 2)

= 28 + 32 + 32

= 92

Total cache size = 92 \* 4 = 368 bits

**Part c:**

AND Gates = 1;

Equality Operator = 1;

2-to-1 Multiplexer = 1;

**Question 3:**

**Part a:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Instruction | Iteration No. | | | | | |
| **1** | **2** | **3** | **4** | **5** |
| lw $t1, 0x24($0) | Compulsory | Capacity | Capacity | Capacity | Capacity |
| lw $t2, 0x2C($0) | Compulsory | Capacity | Capacity | Capacity | Capacity |
| lw $t3, 0x28($0) | Capacity | Capacity | Capacity | Capacity | Capacity |

**Part b:**

number of blocks = 2/1 = 2 words.

number 0f sets = 2 / 2 = 1

Number of set (bits) = log2(20) = 0

Block offset = log2(20) = 0

So, tag bits are 32 – 2 = 30

Total bits = 1 + 2 (ie: V bits) + 2 \* 30 ( ie: Tag bits:2-way associative cache) + 2 \* 32 (data bits:2-way associative cache))

= **127 bits.**

**Part c:**

Equality Comparators = 2.

And Gates = 2.

Mux = 1 (2x1 Mux).

Or gates = 1

**Question 4:**

**AMAT = tL1 + MRL1  (tL2 + MRL2tMM)**

Where tL1 = access time of L1 cache,

tL2 = access time of L2 cache,

tMM = access time of main memory,

MRL1 = L1 miss rate,

MRL2 = L2 miss rate.

So, AMAT in this situation is:

AMAT = 1 + 0.05 ( 4 + 0.25 \* 80) = **2.2 cycles**

So, with 2 GHz clock rate and 1012 instructions to execute, the total execution time is:

Using the clock rate (2 GHz), each clock cycle is: 1/2x109 = 5x10-10 seconds

Total number of cycles using AMAT = 2.2 x 1012 = 22 x 1011 cycles

Total execution time = 22 x 1011 cycles **x** 5x10-10 seconds = **1100 seconds**

**Question 5:**

.text

main:

la $a0, main\_menu

li $v0, 4

syscall

la $a0, input\_option

li $v0, 4

syscall

#input options

li $v0, 5

syscall

beq $v0, 1, option1

beq $v0, 2, option2

beq $v0, 3, option3

beq $v0, 4, option4

beq $v0, 5, option5

beq $v0, 6, option6

beq $v0, 7, option7

beq $v0, 8, option8

j main

option1:

la $a0, opt1

li $v0, 4

syscall

#input options

li $v0, 5

syscall

move $s1, $v0

mul $a0, $s1, $s1

addi $t1, $zero, 4

mul $a0, $a0, $t1

li $v0, 9

syscall

move $s0, $v0

add $t1, $zero, $zero

add $t2, $zero, $zero

add $t3, $zero, $s0

loopInputOuter1:

beq $t1, $s1, loopInput1

addi $t1, $t1, 1

loopInputInner1:

beq $t2, $s1, incrementInput1

addi $t2, $t2, 1

la $a0, input\_value

li $v0, 4

syscall

move $a0, $t1

li $v0, 1

syscall

la $a0, comma

li $v0, 4

syscall

move $a0, $t2

li $v0, 1

syscall

la $a0, endbracket

li $v0, 4

syscall

#input options

li $v0, 5

syscall

sw $v0, 0($t3)

addi $t3, $t3, 4

j loopInputInner1

incrementInput1:

add $t2, $zero, $zero

j loopInputOuter1

loopInput1:

j main

option2:

la $a0, opt1

li $v0, 4

syscall

#input options

li $v0, 5

syscall

move $s1, $v0

mul $a0, $s1, $s1

addi $t1, $zero, 4

mul $a0, $a0, $t1

li $v0, 9

syscall

move $s0, $v0

add $t1, $zero, $zero

add $t2, $zero, $zero

add $t3, $zero, $s0

addi $t4, $zero, 1

loopInputOuter2:

beq $t1, $s1, loopInput2

addi $t1, $t1, 1

loopInputInner2:

beq $t2, $s1, incrementInput2

addi $t2, $t2, 1

sw $t4, 0($t3)

addi $t3, $t3, 4

addi $t4, $t4, 1

j loopInputInner2

incrementInput2:

add $t2, $zero, $zero

j loopInputOuter2

loopInput2:

j main

option3:

la $a0, opt3

li $v0, 4

syscall

#input options

li $v0, 5

syscall

move $t0, $v0

la $a0, opt4

li $v0, 4

syscall

#input options

li $v0, 5

syscall

move $t1, $v0

subi $t0, $t0, 1

subi $t1, $t1, 1

#skipping rows

addi $t5, $zero, 4

mul $t2, $t0, $s1

mul $t3, $t2, $t5

#skipping columns

mul $t2, $t5, $t1

add $t3, $t3, $t2

add $t3, $s0, $t3

lw $t2, 0($t3)

move $a0, $t2

li $v0, 1

syscall

la $a0, newline

li $v0, 4

syscall

j main

option4:

add $t1, $zero, $zero

add $t2, $zero, $zero

add $t3, $zero, $s0

loopInputOuter4:

beq $t1, $s1, loopInput4

loopInputInner4:

beq $t2, $s1, incrementInput4

lw $t4, 0($t3)

move $a0, $t4

li $v0, 1

syscall

addi $t3, $t3, 4

la $a0, space

li $v0, 4

syscall

addi $t2, $t2, 1

j loopInputInner4

incrementInput4:

la $a0, newline

li $v0, 4

syscall

addi $t1, $t1, 1

add $t2, $zero, $zero

j loopInputOuter4

loopInput4:

j main

option5:

add $t0, $zero, $zero

addi $t5, $zero, 4

add $v0, $zero, $zero

loopThrough5:

beq $t0, $s1, endOption5

#skipping rows

mul $t1, $t0, $s1

mul $t2, $t1, $t5

#skipping columns

mul $t3, $t5, $t0

add $t4, $t3, $t2

add $t3, $s0, $t4

lw $t4, 0($t3)

add $v0, $v0, $t4

addi $t0, $t0, 1

j loopThrough5

endOption5:

move $t0, $v0

la $a0, option5\_prompt

li $v0, 4

syscall

move $a0, $t0

li $v0, 1

syscall

la $a0, newline

li $v0, 4

syscall

j main

option6:

add $t0, $s1, $zero

addi $t5, $zero, 4

add $v0, $zero, $zero

loopThrough6:

beq $t0, $zero, endOption6

subi $t0, $t0, 1

#skipping rows

mul $t1, $t0, $s1

mul $t2, $t1, $t5

#skipping columns

mul $t3, $t5, $t0

add $t4, $t3, $t2

add $t3, $s0, $t4

lw $t4, 0($t3)

add $v0, $v0, $t4

j loopThrough6

endOption6:

move $t0, $v0

la $a0, option6\_prompt

li $v0, 4

syscall

move $a0, $t0

li $v0, 1

syscall

la $a0, newline

li $v0, 4

syscall

j main

option7:

add $t0, $zero, $zero

add $t1, $zero, $zero

add $t2, $zero, $zero

addi $t5, $zero, 4

loopInputOuter7:

beq $t0, $s1, loopInput7

loopInputInner7:

beq $t1, $s1, incrementInput7

#skipping rows

mul $t3, $t0, $s1

mul $t4, $t3, $t5

#skipping columns

mul $t3, $t5, $t1

add $t3, $t3, $t4

add $t3, $s0, $t3

lw $t4, 0($t3)

addi $t3, $t3, 4

add $t2, $t2, $t4

addi $t1, $t1, 1

j loopInputInner7

incrementInput7:

add $t1, $zero, $zero

addi $t0, $t0, 1

j loopInputOuter7

loopInput7:

la $a0, option7\_prompt

li $v0, 4

syscall

move $a0, $t2

li $v0, 1

syscall

la $a0, newline

li $v0, 4

syscall

j main

option8:

add $t0, $zero, $zero

add $t1, $zero, $zero

add $t2, $zero, $zero

addi $t5, $zero, 4

loopInputOuter8:

beq $t0, $s1, loopInput8

loopInputInner8:

beq $t1, $s1, incrementInput8

#skipping columns

mul $t3, $t1, $s1

mul $t4, $t3, $t5

#skipping rows

mul $t3, $t5, $t0

add $t3, $t3, $t4

add $t3, $s0, $t3

lw $t4, 0($t3)

addi $t3, $t3, 4

add $t2, $t2, $t4

addi $t1, $t1, 1

j loopInputInner8

incrementInput8:

add $t1, $zero, $zero

addi $t0, $t0, 1

j loopInputOuter8

loopInput8:

la $a0, option8\_prompt

li $v0, 4

syscall

move $a0, $t2

li $v0, 1

syscall

la $a0, newline

li $v0, 4

syscall

j main

.data

main\_menu: .asciiz "1. Ask the user the matrix size in terms of its dimensions (N), and then ask the user enter matrix elements row by row.\n2. Ask the user the matrix size in terms of its dimensions (N), and initialize the matrix entries with consecutive values (1, 2, 3 ...) \n3. Display a desired element of the matrix by specifying its row and column number, \n4. Display entire matrix row by row \n5. Obtain trace of the matrix and display,\n6. Obtain trace like summation using the other diagonal of the matrix and display,\n7. Obtain sum of matrix elements by row-major (row by row) summation,\n8. Obtain sum of matrix elements by column-major (column by column) summation."

input\_option: .asciiz "\nPlease select your option: "

input\_value: .asciiz "Enter value for ("

comma: .asciiz ", "

endbracket: .asciiz "):"

space: .asciiz " "

newline: .asciiz "\n"

opt1: .asciiz "Enter matrix size in dimensions (N): "

opt3: .asciiz "Please enter row: "

opt4: .asciiz "Please enter column: "

option5\_prompt: .asciiz "The trace is: "

option6\_prompt: .asciiz "The reverse trace is: "

option7\_prompt: .asciiz "Row-by-row sum is: "

option8\_prompt: .asciiz "column-by-column sum is: "